数据库之MySQL

华清远见-成都中心-H5教学部

目录

第34章 数据库基本概念

华清远见-成都中心-H5教学部

数据库基本概念

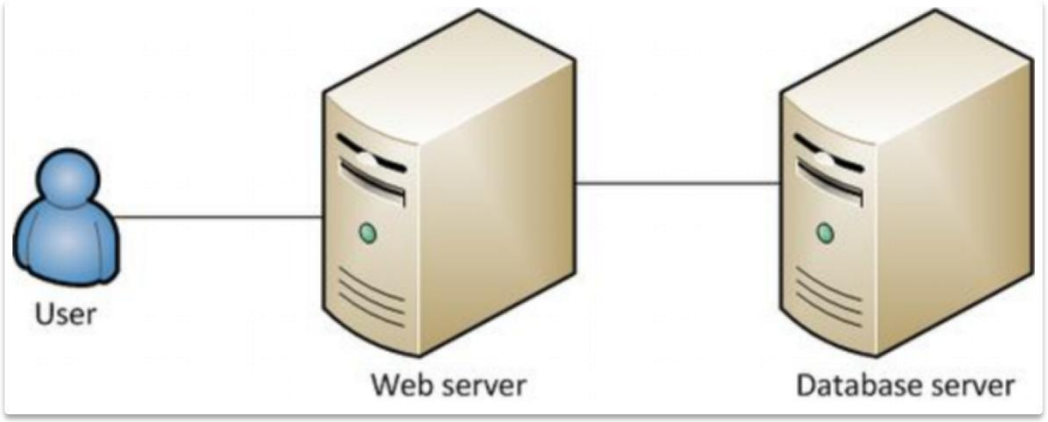
目录

- 数据库服务器、数据库管理系统
- 常见的数据库管理系统
- 数据库和表
- 记录、字段/列、数据
- 登录和管理数据库管理系统

数据库服务器、数据库管理系统

一． 数据库服务器：

安装了数据库管理系统软件的计算机叫数据库服务器。



二． 数据库管理系统：

简称DBMS(DataBase Management System)是用来管理数据库的软件，用于建立、使用和维护数据库，对数据库进行统一管理和控制，以保证数据库的安全性和完整性。

RDBMS 指的是关系型数据库管理系统，对应的则是非关系型数据库。

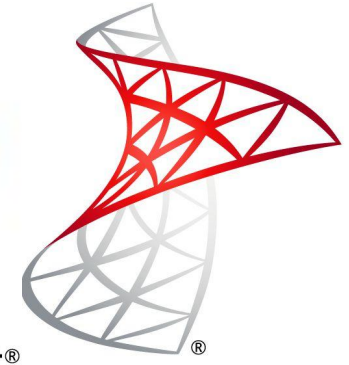
常见的数据库管理系统

一． Oracle：RDBMS，甲骨文公司的一款关系数据库管理系统，一般作为大型或超大型网站的数据库。

二． SQLServer： RDBMS，微软公司的。

三． MySQL： RDBMS， MySQL由瑞典MySQL AB公司开发，后被Oracle公司收购。

四． MongoDB：适用于敏捷开发的非关系型数据库。



Microsoft®
SQL Server® 2012

数据库和表

一． 数据库

数据库（Database）是按照数据结构来组织、存储和管理数据的仓库。

二． 数据表

数据表是数据库中存放数据的地方，是相关的数据项的集合，它由列（字段）和行（记录）组成；

一个数据库中可包含若干个数据表；

数据库只是一个袋子，装在里面的数据表是内容。

记录、字段/列、数据

一． 记录、字段/列、数据

记录：表中的行叫记录，通常用它所在的行数(id值)表示这是第几条记录；

字段：一条记录由多列构成，每个列称为字段，有些时候也叫属性；

数据：存放在表行（记录）列（字段）交叉处的值叫数据。

二． 数据库服务器、数据库、表、记录、字段关系的通俗理解

如果把	数据库服务器	看成	文件柜

==========

那么	数据库管理系统	就是	文件柜中的格子
	数据库	就是	格子中的文件袋
	表	就是	文件袋中的表
	字段/列	就是	表头
	记录	就是	表头下面的一行一行的数据

登录和管理数据库管理系统

一、直接命令行登录：

mysql –h主机名 –u用户名 –p

回车，然后输入密码，回车进入数据库管理命令行。

二、Navicat客户端：非常流行和专业的客户端，收费，免费的认证码有限。

三、HeidiSQL 客户端：免费，小巧轻便，配置和操作简单，推荐使用。

第35章 SQL基础

华清远见-成都中心-H5教学部

SQL基础

目录

SQL简介及语法

一．什么是SQL

SQL 是用于访问和处理数据库的标准的计算机语言：

① SQL 指结构化查询语言；

② SQL 使我们有能力访问和操作数据库；

③ SQL 是一种 ANSI 的标准计算机语言。

编者注：ANSI，美国国家标准化组织

二．SQL 能做什么？

① SQL 面向数据库执行查询；

② SQL 可从数据库取回数据；

③ SQL 可在数据库中插入新的记录；

④ SQL 可更新数据库中的数据；

⑤ SQL 可从数据库删除记录；

⑥ SQL 可创建新数据库；

⑦ SQL 可在数据库中创建新表。

三．SQL 是一种标准

SQL 是一门 ANSI 的标准计算机语言，用来访问和操作数据库系统。SQL 语句用于取回和更新数据库中的数据。SQL 可与数据库程序协同工作，比如 MySQL、DB2、Informix、MS SQL Server、Oracle、Sybase 以及其他数据库系统。

不幸地是，存在着很多不同版本的 SQL 语言，但是为了与 ANSI 标准相兼容，它们必须以相似的方式共同地来支持一些主要的关键词（比如 SELECT、UPDATE、DELETE、INSERT、WHERE 等等）。

注释：除了 SQL 标准之外，大部分 SQL 数据库程序都拥有它们自己的私有扩展。

四．数据库表

一个数据库通常包含一个或多个表。每个表由一个名字标识（例如“members”或者“students”）。表包含带有数据的记录（行）。

下面的例子是一个名为 "Persons" 的表：

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

上面的表包含三条记录（每一条对应一个人）和五个列（Id、姓、名、地址和城市）。

SQL简介及语法

五．SQL 语句

你需要在数据库上执行的大部分工作都由 SQL 语句完成。

下面的语句从表中选取Id和LastName列的数据：

```
SELECT Id, LastName FROM Persons
```

结果集类似这样：

Id	LastName
1	Adams
2	Bush
3	Carter

SQL简介及语法

六．SQL 语句后面的分号

某些数据库系统要求在每条 SQL 命令的末端使用分号。分号是在数据库系统中分隔SQL 语句的标准方法，这样就可以在对服务器的相同请求中执行一条以上的语句。

七．DML 和 DDL

可以把 SQL 分为两个部分：数据操作语言 (DML) 和数据定义语言 (DDL)；

SQL (结构化查询语言)是用于执行查询的语法，也包含用于更新、插入和删除记录的语法。

SQL简介及语法

八．DML 和 DDL常用语句

查询和更新指令构成了 SQL 的 DML 部分：

① SELECT - 从数据库表中获取数据；

② UPDATE - 更新数据库表中的数据；

③ DELETE - 从数据库表中删除数据；

④ INSERT INTO - 向数据库表中插入数据；

⑤ SQL 的数据定义语言 (DDL) 部分使我们有能力创建或删除表格。

SQL 中最重要的 DDL 语句:

① CREATE DATABASE - 创建新数据库；

② ALTER DATABASE - 修改数据库；

③ CREATE TABLE - 创建新表；

④ ALTER TABLE - 变更（改变）数据库表；

⑤ DROP TABLE - 删除表。

SELECT 语句

SELECT 语句用于从表中选取数据，结果存储在一个结果表中（称为结果集）。

一．SELECT 语法

```
SELECT 列名称 FROM 表名称
```

以及：

```
SELECT * FROM 表名称
```

注释：SQL 语句对大小写不敏感，SELECT 等效于 select。

SELECT 语句

二．SELECT 实例

如需从名为 “Persons”的表获取 “LastName” 和 “FirstName” 的列的内容，使用如下SELECT 语句：

```
SELECT LastName, FirstName FROM Persons
```

执行结果如下：

LastName	FirstName
Adams	John
Bush	George
Carter	Thomas

SELECT 语句

三．SELECT * 实例

现在我们希望从 “Persons” 表中选取所有的列，请使用符号 * 取代列的名称，就像这样：

```
SELECT * FROM Persons
```

提示：星号（*）是选取所有列的快捷方式。

执行结果如下：

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

WHERE 子句

如需有条件地从表中选取数据，可将 WHERE 子句添加到 SELECT 语句。

一．语法

```
SELECT 列名称 FROM 表名称 WHERE 列运算符值
```

下面的运算符可在 WHERE 子句中使用：

操作符	描述
=	等于
<> （或 != ）	不等于
>	大于
<	小于
>=	大于等于
<=	小于等于
BETWEEN AND	在某个范围内
LIKE	搜索某种模式

WHERE 子句

二．使用 WHERE 子句

如果只希望选取居住在城市 "Beijing" 中的人，我们需要向 SELECT 语句添加 WHERE 子句：

```
SELECT * FROM Persons WHERE City='Beijing'
```

提示：星号（*）是选取所有列的快捷方式。

执行结果如下：

Id	LastName	FirstName	Address	City
3	Carter	Thomas	Changan Street	Beijing

WHERE 子句

三．引号的使用

请注意，我们在例子中的条件值周围使用的是单引号。

SQL 使用单引号来环绕文本值（大部分数据库系统也接受双引号）。如果是数值，不需要使用引号。

这是正确的：

```
SELECT * FROM Persons WHERE FirstName='Bush'
```

这是错误的：

```
SELECT * FROM Persons WHERE FirstName=Bush
```

AND 和 OR 运算符

AND 和 OR 可在 WHERE 子语句中把两个或多个条件结合起来。

如果第一个条件和第二个条件都成立，则 AND 运算符显示一条记录。

如果第一个条件和第二个条件中只要有一个成立，则 OR 运算符显示一条记录。

原始表：

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing
4	Carter	William	TianAnMen Street	Beijing

AND 和 OR 运算符

一．AND 运算符实例

使用 AND 来显示所有姓为 "Carter" 并且名为 "Thomas" 的人：

```
SELECT * FROM Persons WHERE FirstName='Thomas' AND
LastName='Carter'
```

查询结果：

Id	LastName	FirstName	Address	City
3	Carter	Thomas	Changan Street	Beijing

AND 和 OR 运算符

二．OR 运算符实例

使用 OR 来显示所有姓为 "Carter" 或者名为 "Thomas" 的人：

```
SELECT * FROM Persons WHERE firstname='Thomas' OR
lastname='Carter'
```

查询结果：

Id	LastName	FirstName	Address	City
3	Carter	Thomas	Changan Street	Beijing
4	Carter	William	TianAnMen Street	Beijing

AND 和 OR 运算符

三．结合 AND 和 OR 运算符实例

我们也可以把 AND 和 OR 结合起来（使用圆括号来组成复杂的表达式）：

```
SELECT * FROM Persons WHERE FirstName='Thomas' OR
(FirstName='William' AND LastName='Carter')
```

查询结果：

Id	LastName	FirstName	Address	City
3	Carter	Thomas	Changan Street	Beijing
4	Carter	William	TianAnMen Street	Beijing

ORDER BY 语句

ORDER BY 语句用于根据指定的列对结果集进行排序。

ORDER BY 语句默认按照升序对记录进行排序。

如果你希望按照降序对记录进行排序，可以使用 DESC 关键字。

原始表：

Company	OrderNumber
IBM	3532
FarSight	2356
Apple	4698
FarSight	6953

ORDER BY 语句

一． 实例 1

以字母顺序显示公司名称：

```
SELECT Company, OrderNumber FROM Orders ORDER BY Company
```

结果：

Company	OrderNumber
Apple	4698
FarSight	6953
FarSight	2356
IBM	3532

ORDER BY 语句

二． 实例 2

以字母顺序显示公司名称（Company），并以数字顺序显示顺序号（OrderNumber）：

```
SELECT Company, OrderNumber FROM Orders ORDER BY Company,
OrderNumber
```

结果：

Company	OrderNumber
Apple	4698
FarSight	2356
FarSight	6953
IBM	3532

ORDER BY 语句

三． 实例 3

以逆字母顺序显示公司名称：

```
SELECT Company, OrderNumber FROM Orders ORDER BY Company DESC
```

结果：

Company	OrderNumber
IBM	3532
FarSight	6953
FarSight	2356
Apple	4698

ORDER BY 语句

四． 实例 4

以逆字母顺序显示公司名称，并以数字顺序显示顺序号：

```
SELECT Company, OrderNumber FROM Orders ORDER BY Company DESC,
OrderNumber ASC
```

结果：

Company	OrderNumber
IBM	3532
FarSight	2356
FarSight	6953
Apple	4698

SELECT DISTINCT 语句

在表中，可能会包含重复值。这并不成问题，不过，有时你也许希望仅仅列出不同（distinct）的值。

关键词 DISTINCT 用于返回唯一不同的值。

一． 语法

```
SELECT DISTINCT 列名称 FROM 表名称
```

二．使用 DISTINCT 关键词

如果要从 "Company" 列中选取所有的值，我们需要使用 SELECT 语句：

```
SELECT Company FROM Orders
```

此时，在结果集中，FarSight 被列出了两次。

SELECT DISTINCT 语句

如需从 Company" 列中仅选取唯一不同的值，我们需要使用 SELECT DISTINCT 语句：

```
SELECT DISTINCT Company FROM Orders
```

结果：

Company
IBM
FarSight
Apple

现在，在结果集中，“FarSight” 仅被列出了一次。

INSERT INTO 语句

INSERT INTO 语句用于向表格中插入新的行。

一．语法

```
INSERT INTO 表名称 VALUES (值 1, 值 2,....)
```

我们也可以指定所要插入数据的列：

```
INSERT INTO 表名称(列 1, 列 2,...) VALUES (值 1, 值 2,....)
```

"Persons" 表：

Id	LastName	FirstName	Address	City
1	Carter	Thomas	Changan Street	Beijing

INSERT INTO 语句

二．插入新的行

SQL 语句：

```
INSERT INTO Persons VALUES ('Gates', 'Bill', 'Xuanwumen 10',
'Beijing')
```

"Persons" 表：

Id	LastName	FirstName	Address	City
1	Carter	Thomas	Changan Street	Beijing
2	Gates	Bill	Xuanwumen 10	Beijing

INSERT INTO 语句

三．在指定的列中插入数据

SQL 语句：

```
INSERT INTO Persons (LastName, Address) VALUES ('Wilson',
'Champs-Elysees')
```

"Persons" 表：

Id	LastName	FirstName	Address	City
1	Carter	Thomas	Changan Street	Beijing
2	Gates	Bill	Xuanwumen 10	Beijing
3	Wilson		Champs-Elysees	

Update 语句

Update 语句用于修改表中的数据。

一．语法

```
UPDATE 表名称 SET 列名称 = 新值 WHERE 列名称 = 某值
```

"Persons" 表：

Id	LastName	FirstName	Address	City
1	Gates	Bill	Xuanwumen 10	Beijing
2	Wilson		Champs-Elysees	

Update 语句

二．更新某一行中的一个列

我们为lastname是 "Wilson" 的人添加firstname：

```
UPDATE Person SET FirstName = 'Fred' WHERE LastName = 'Wilson'
```

更新后"Persons" 表：

Id	LastName	FirstName	Address	City
1	Gates	Bill	Xuanwumen 10	Beijing
2	Wilson	Fred	Champs-Elysees	

Update 语句

三．更新某一行中的若干列

我们会修改地址（address），并添加城市名称（city）：

```
UPDATE Person SET Address = 'Zhongshan 23', City = 'Nanjing'
WHERE LastName = 'Wilson'
```

更新后"Persons" 表：

Id	LastName	FirstName	Address	City
1	Gates	Bill	Xuanwumen 10	Beijing
2	Wilson	Fred	Zhongshan 23	Nanjing

DELETE 语句

DELETE 语句用于删除表中的行。

一．语法

```
DELETE FROM 表名称 WHERE 列名称 = 值
```

"Persons" 表：

Id	LastName	FirstName	Address	City
1	Gates	Bill	Xuanwumen 10	Beijing
2	Wilson	Fred	Zhongshan 23	Nanjing

DELETE 语句

二．删除某行

"Fred Wilson" 会被删除：

```
DELETE FROM Person WHERE LastName = 'Wilson'
```

"Persons" 表：

Id	LastName	FirstName	Address	City
1	Gates	Bill	Xuanwumen 10	Beijing

DELETE 语句

三．删除所有行

可以在不删除表的情况下删除所有的行。这意味着表的结构、属性和索引都是完整的：

```
DELETE FROM table_name
```

通配符及LIKE 操作符

在搜索数据库中的数据时，SQL 通配符可以替代一个或多个字符。SQL 通配符必须与 LIKE 运算符一起使用。

在 SQL 中，可使用以下通配符：

通配符	描述
%	替代一个或多个字符
_	仅替代一个字符

"Persons" 表：

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

通配符及LIKE 操作符

一．使用 % 通配符

例子

现在，我们希望从上面的 "Persons" 表中选取居住在以 "Ne" 开始的城市里的人：

我们可以使用下面的 SELECT 语句：

```
SELECT * FROM Persons WHERE City LIKE 'Ne%'
```

“Persons” 表的查询结果集：

Id	LastName	FirstName	Address	City
2	Bush	George	Fifth Avenue	New York

通配符及LIKE 操作符

二．使用 _ 通配符

例子1

现在，我们希望从上面的 "Persons" 表中选取名字的第一个字符之后是 "eorge" 的人：

我们可以使用下面的 SELECT 语句：

```
SELECT * FROM Persons WHERE FirstName LIKE '_eorge'
```

“Persons” 表的查询结果集：

Id	LastName	FirstName	Address	City
2	Bush	George	Fifth Avenue	New York

通配符及LIKE 操作符

例子2

接下来，我们希望从 "Persons" 表中选取的这条记录的姓氏以 "C" 开头，然后是一个任意字符，然后是 "r"，然后是一个任意字符，然后是 "er"：

我们可以使用下面的 SELECT 语句：

```
SELECT * FROM Persons
WHERE LastName LIKE 'C_r_er'
```

“Persons” 表的查询结果集：

Id	LastName	FirstName	Address	City
3	Carter	Thomas	Changan Street	Beijing

通配符及LIKE 操作符

三．LIKE 操作符

LIKE 操作符用于在 WHERE 子句中搜索列中的指定模式。

LIKE 操作符语法

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern
```

通配符及LIKE 操作符

四．LIKE 操作符实例

例子 1

现在，我们希望从上面的 "Persons" 表中选取居住在以 "N" 开始的城市里的人：

我们可以使用下面的 SELECT 语句：

```
SELECT * FROM Persons WHERE City LIKE 'N%'
```

提示："%" 可用于定义通配符（模式中缺少的字母）。

“Persons” 表的查询结果集：

Id	LastName	FirstName	Address	City
2	Bush	George	Fifth Avenue	New York

通配符及LIKE 操作符

例子 2

接下来，我们希望从 "Persons" 表中选取居住在以 "g" 结尾的城市里的人：

我们可以使用下面的 SELECT 语句：

```
SELECT * FROM Persons WHERE City LIKE '%g'
```

“Persons” 表的查询结果集：

Id	LastName	FirstName	Address	City
3	Carter	Thomas	Changan Street	Beijing

通配符及LIKE 操作符

例子 3

接下来，我们希望从 "Persons" 表中选取居住在包含 "lon" 的城市里的人：

我们可以使用下面的 SELECT 语句：

```
SELECT * FROM Persons WHERE City LIKE '%lon%'
```

“Persons” 表的查询结果集：

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London

通配符及LIKE 操作符

例子 4

通过使用 NOT 关键字，我们可以从 "Persons" 表中选取居住在不包含 "lon" 的城市里的人：

我们可以使用下面的 SELECT 语句：

```
SELECT * FROM Persons WHERE City NOT LIKE '%lon%'
```

“Persons” 表的查询结果集：

Id	LastName	FirstName	Address	City
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

LIMIT 子句

limit 子句用于规定要返回的记录的数目。对于拥有数千条记录的大型表来说，limit子句是非常有用的。

一．语法

```
SELECT column_name(s) FROM table_name LIMIT start, number
```

注释：start表示开始位置，0表示第一条，number表示取出来的数据数量。

"Persons" 表：

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing
4	Obama	Barack	Pennsylvania Avenue	Washington

LIMIT 子句

二．实例

```
SELECT * FROM Persons LIMIT 1, 2
```

"Persons" 表：

Id	LastName	FirstName	Address	City
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

IN 操作符

IN 操作符允许我们在 WHERE 子句中规定多个值。

一．语法

```
SELECT column_name(s) FROM table_name
WHERE column_name IN (value1, value2, ...)
```

"Persons" 表：

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

IN 操作符

二．IN 操作符实例

现在，我们希望从上表中选取姓氏为 Adams 和 Carter 的人：

我们可以使用下面的 SELECT 语句：

```
SELECT * FROM Persons
WHERE LastName IN ('Adams','Carter')
```

“Persons” 表的查询结果集：

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
3	Carter	Thomas	Changan Street	Beijing

BETWEEN 操作符

操作符 BETWEEN ... AND 会选取介于两个值之间的数据范围。这些值可以是数值、文本或者日期。

一．语法

```
SELECT column_name(s) FROM table_name
WHERE column_name BETWEEN value1 AND value2
```

"Persons" 表：

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

BETWEEN 操作符

二．语法

```
SELECT * FROM Persons  WHERE Id BETWEEN 2 AND 3
```

"Persons" 表的查询结果集：

Id	LastName	FirstName	Address	City
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

第36章 SQL高级

华清远见-成都中心-H5教学部

SQL高级

目录

JOIN连接

用于根据两个或多个表中的列之间的关系，从这些表中查询数据。

一．Join 和 Key

有时为了得到完整的结果，我们需要从两个或更多的表中获取结果。我们就需要执行 join。数据库中的表可通过键将彼此联系起来。主键（Primary Key）是一个列，在这个列中的每一行的值都是唯一的。在表中，每个主键的值都是唯一的。这样做的目的是在不重复每个表中的所有数据的情况下，把表间的数据交叉捆绑在一起。

"Persons" 表：

Id_P	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

请注意，"Id_P" 列是 Persons 表中的的主键。这意味着没有两行能够拥有相同的Id_P。即使两个人的姓名完全相同，Id_P也可以区分他们。

JOIN连接

"Orders" 表：

Id_O	OrderNo	Id_P
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	65

请注意，“Id_O” 列是 Orders 表中的的主键，同时，“Orders” 表中的 “Id_P” 列用于引用 "Persons" 表中的人，而无需使用他们的确切姓名。

请留意，"Id_P" 列把上面的两个表联系了起来。

JOIN连接

二．引用两个表

我们可以通过引用两个表的方式，从两个表中获取数据：谁订购了产品，并且他们订购了什么产品？

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons, Orders WHERE Persons.Id_P = Orders.Id_P
```

查询结果集：

LastName	FirstName	OrderNo
Adams	John	22456
Adams	John	24562
Carter	Thomas	77895
Carter	Thomas	44678

JOIN连接

三．使用JOIN

除了上面的方法，我们也可以使用关键词 JOIN 来从两个表中获取数据。

如果我们希望列出所有人的定购，可以使用下面的 SELECT 语句：

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo FROM
Persons INNER JOIN Orders ON Persons.Id_P = Orders.Id_P
ORDER BY Persons.LastName
```

"Persons" 表：

LastName	FirstName	OrderNo
Adams	John	22456
Adams	John	24562
Carter	Thomas	77895
Carter	Thomas	44678

JOIN连接

四．不同的 JOIN

下面列出了您可以使用的 JOIN 类型，以及它们之间的差异：

① INNER JOIN: 内连接，只有两个表都匹配才会显示记录。

② LEFT JOIN: 左（外）连接，即使右表中没有匹配，也从左表返回所有的行。

③ RIGHT JOIN: 右（外）连接，即使左表中没有匹配，也从右表返回所有的行。

④ FULL JOIN: 全连接，只要其中一个表中存在匹配，就返回行。

JOIN连接

五．FULL JOIN 关键字

只要其中某个表存在匹配，FULL JOIN 关键字就会返回行。mysql中没有FULL JOIN,可以采用左连接+ union+右连接的 方式来实现。

UNION 操作符用于合并两个或多个 SELECT 语句的结果集。UNION 内部的 SELECT 语句必须拥有相同数量的列。列也必须拥有相似的数据类型。同时，每条 SELECT 语句中的列的顺序必须相同。

列出所有的人，以及他们的定单，以及所有的定单，以及定购它们的人。您可以使用下面的 SELECT 语句：

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons LEFT JOIN Orders ON Persons.Id_P=Orders.Id_P
UNION
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons RIGHT JOIN Orders ON Persons.Id_P=Orders.Id_P
```

JOIN连接

FULL JOIN 查询结果集：

LastName	FirstName	OrderNo
Adams	John	22456
Adams	John	24562
Carter	Thomas	77895
Carter	Thomas	44678
Bush	George	
		34764

注意：LEFT JOIN会从左表中返回所有行，RIGHT JOIN会从右表中返回所有行，UNION将两个结果集进行合并。

CREATE TABLE 语句

CREATE TABLE 语句用于创建数据库中的表。

一. CREATE TABLE 语法

```
CREATE TABLE 表名称 (
列名称 1 数据类型,
列名称 2 数据类型,
列名称 3 数据类型,
....)
```

CREATE TABLE 语句

数据类型（data_type）规定了列可容纳何种数据类型。下面的表格包含了SQL中最常用的数据类型：

数据类型	描述
➢ integer(size) ➢ int(size) ➢ smallint(size) ➢ tinyint(size)	仅容纳整数。在括号内规定数字的最大位数。
➢ decimal(size,d) ➢ numeric(size,d)	容纳带有小数的数字。"size" 规定数字的最大位数。"d" 规定小数点右侧的最大位数。
char(size)	容纳固定长度的字符串（可容纳字母、数字以及特殊字符）。在括号中规定字符串的长度。
varchar(size)	容纳可变长度的字符串（可容纳字母、数字以及特殊的字符）。在括号中规定字符串的最大长度。
date(yyyymmdd)	容纳日期。

CREATE TABLE 语句

二．CREATE TABLE 实例

本例演示如何创建名为 "Person" 的表。该表包含 5 个列，列名分别是："Id_P"、"LastName"、"FirstName"、"Address" 以及 "City"：

```
CREATE TABLE Persons (
    Id_P int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
)
```

Id_P列的数据类型是int，包含整数。其余 4 列的数据类型是varchar。

PRIMARY KEY 约束

PRIMARY KEY 约束唯一标识数据库表中的每条记录。

① 主键必须包含唯一的值。

② 主键列不能包含 NULL 值。

③ 每个表都应该有一个主键，并且每个表只能有一个主键。

PRIMARY KEY 约束

一．建表时的主键约束

下面的 SQL 在 "Persons" 表创建时在 "Id_P" 列创建 PRIMARY KEY 约束：

```
CREATE TABLE Persons (

Id_P int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255),
PRIMARY KEY (Id_P)
)
```

AUTO INCREMENT 语句

Auto-increment 会在新记录插入表中时生成一个唯一的数字。

我们通常希望在每次插入新记录时，自动地创建主键字段的值。

我们可以在表中创建一个 auto-increment 字段。

下列 SQL 语句把 "Persons" 表中的 "P_Id" 列定义为 auto-increment 主键：

```
CREATE TABLE Persons (
        P_Id int NOT NULL AUTO_INCREMENT,
        LastName varchar(255) NOT NULL,
        FirstName varchar(255),
        Address varchar(255),
        City varchar(255),
        PRIMARY KEY (P_Id)
)
```

AUTO INCREMENT 语句

MySQL 使用 AUTO_INCREMENT 关键字来执行 auto-increment 任务。

默认地，AUTO_INCREMENT 的开始值是 1，每条新记录递增 1。

要在 "Persons" 表中插入新记录，我们不必为 "P_Id" 列规定值（会自动添加一个唯一的值）：

```
INSERT INTO Persons (FirstName, LastName)
VALUES ('Bill', 'Gates')
```

上面的 SQL 语句会在 "Persons" 表中插入一条新记录。"P_Id" 会被赋予一个唯一的值。"FirstName" 会被设置为 "Bill"，"LastName" 列会被设置为 "Gates"。

SQL函数

Mysql 拥有很多可用于计数和计算的内建函数。

内建 SQL 函数的语法是：

```
SELECT function(列) FROM 表
```

在 SQL 中，基本的函数类型和种类有若干种。函数的基本类型是：合计函数和标量函数。

合计函数

合计函数的操作面向一系列的值，并返回一个单一的值。如：

① COUNT()函数：用来统计记录的条数；

② SUM()函数:是求和函数；

③ AVG()函数:是求平均值的函数；

④ MAX()函数是求最大值的函数；

⑤ MIN()函数是求最小值的函数；

注释：group by 语句经常和合计函数配合使用来进行求和、计数、求平均值、求最大值和求最小值。

SQL函数

标量函数

标量函数的操作面向某个单一的值，并返回基于输入值的一个单一的值。如：

① abs：该函数返回一个数值表达式的绝对值。如abs(-123)；

② addtime：把两个时间表达式加起来。如addtime(‘100:00:00’，‘200:02:04’)；

③ concat：该函数合并两个字符串的值；

④ convert：该函数转换参数1的数据类型为参数2指定的类型。如：convert(‘12.56’,unsigned integer)；

注释：group by 语句经常和合计函数配合使用来进行求和、计数、求平均值、求最大值和求最小值。

SQL函数

一．AVG 函数

AVG 函数返回数值列的平均值。NULL 值不包括在计算中。

1. AVG()语法

```
SELECT AVG(column_name) FROM table_name
```

"Orders" 表：

O_Id	OrderDate	OrderPrice	Customer
1	2018/12/29	1000	Bush
2	2018/11/23	1600	Carter
3	2018/10/05	700	Bush
4	2018/09/28	300	Bush
5	2018/08/06	2000	Adams
6	2018/07/21	100	Carter

SQL函数

2. AVG()实例

例子 1：

现在，我们希望计算 "OrderPrice" 字段的平均值。

我们使用如下 SQL 语句：

```
SELECT AVG(OrderPrice) AS OrderAverage FROM Orders
```

查询结果集：

OrderAverage
950

SQL函数

例子 2：

现在，我们希望找到 OrderPrice 值高于 OrderPrice 平均值的客户。

我们使用如下 SQL 语句：

```
SELECT Customer FROM Orders
WHERE OrderPrice>(SELECT AVG(OrderPrice) FROM Orders)
```

查询结果集：

Customer
Bush
Carter
Adams

二．COUNT() 函数

COUNT() 函数返回匹配指定条件的行数。

1. COUNT()语法

① COUNT(column_name) 语法

COUNT(column_name) 函数返回指定列的值的数目（NULL 不计入）：

```
SELECT COUNT(column_name) FROM table_name
```

② COUNT(*) 语法

COUNT(*) 函数返回表中的记录数：

```
SELECT COUNT(*) FROM table_name
```

SQL函数

③ COUNT(DISTINCT column_name) 语法

COUNT(DISTINCT column_name) 函数返回指定列的不同值的数目：

```
SELECT COUNT(DISTINCT column_name) FROM table_name
```

SQL函数

2. COUNT()实例

例子 1：
现在，我们希望计算客户 "Carter" 的订单数。
我们使用如下 SQL 语句：

```
SELECT COUNT(Customer) AS CustomerNilsen FROM Orders
WHERE Customer='Carter'
```

以上 SQL 语句的结果是 2，因为客户 Carter 共有 2 个订单：

查询结果集：

CustomerNilsen
2

SQL函数

例子 2：

现在，我们希望计算 "Orders" 表中不同客户的数目。

我们使用如下 SQL 语句：

```
SELECT COUNT(DISTINCT Customer) AS NumberOfCustomers FROM Orders
```

这是 "Orders" 表中不同客户（Bush, Carter 和 Adams）的数目。

查询结果集：

NumberOfCustomers
3

GROUP BY 语句

合计函数 (比如 SUM) 常常需要添加 GROUP BY 语句。

GROUP BY 语句用于结合合计函数，根据一个或多个列对结果集进行分组。

一．GROUP BY 语法

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
```

GROUP BY 语句

二．GROUP BY 实例

AVG 函数返回数值列的平均值。NULL 值不包括在计算中。

"Orders" 表：

O_Id	OrderDate	OrderPrice	Customer
1	2018/12/29	1000	Bush
2	2018/11/23	1600	Carter
3	2018/10/05	700	Bush
4	2018/09/28	300	Bush
5	2018/08/06	2000	Adams
6	2018/07/21	100	Carter

GROUP BY 语句

现在，我们希望查找每个客户的总金额（总订单）。

我们想要使用 GROUP BY 语句对客户进行组合。我们使用下列 SQL 语句：

```
SELECT Customer,SUM(OrderPrice) FROM Orders
GROUP BY Customer
```

查询结果集：

Customer	SUM(OrderPrice)
Bush	2000
Carter	1700
Adams	2000
Bush	2000

HAVING 子句

在 SQL 中增加 HAVING 子句原因是，WHERE 关键字无法与合计函数一起使用。

一．HAVING语法

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
HAVING aggregate_function(column_name) operator value
```

HAVING 子句

二．HAVING实例

现在，我们希望查找订单总金额少于 2000 的客户。我们使用如下 SQL 语句：

```
SELECT Customer,SUM(OrderPrice) FROM Orders
GROUP BY Customer
HAVING SUM(OrderPrice)<2000
```

查询结果集：

Customer	SUM(OrderPrice)
Carter	1700

HAVING 子句

现在我们希望查找客户 "Bush" 或 "Adams" 拥有超过 1500 的订单总金额。

我们在 SQL 语句中增加了一个普通的 WHERE 子句：

```
SELECT Customer,SUM(OrderPrice) FROM Orders
WHERE Customer='Bush' OR Customer='Adams'
GROUP BY Customer
HAVING SUM(OrderPrice)>1500
```

查询结果集：

Customer	SUM(OrderPrice)
Bush	2000
Adams	2000

MySQL 数据类型

在 MySQL 中，有三种主要的类型：文本、数字和日期/时间类型。

一．Text 类型：

数据类型	描述
CHAR(size)	保存固定长度的字符串（可包含字母、数字以及特殊字符）。在括号中指定字符串的长度。最多 255 个字符。
VARCHAR(size)	保存可变长度的字符串（可包含字母、数字以及特殊字符）。在括号中指定字符串的最大长度。最多 65535个字符。
TINYTEXT	存放最大长度为 255 个字符的字符串。
TEXT	存放最大长度为 65,535 个字符的字符串。
MEDIUMTEXT	存放最大长度为 16,777,215 个字符的字符串。

MySQL 数据类型

二．Number 类型：

数据类型	描述
TINYINT(size)	-128 到 127 常规。0 到 255 无符号*。在括号中规定最大位数。
SMALLINT(size)	-32768 到 32767 常规。0 到 65535 无符号*。在括号中规定最大位数。
MEDIUMINT(size)	-8388608 到 8388607 普通。0 to 16777215 无符号*。在括号中规定最大位数。
INT(size)	-2147483648 到 2147483647 常规。0 到 4294967295 无符号*。在括号中规定最大位数。
BIGINT(size)	-9223372036854775808 到 9223372036854775807 常规。0 到 18446744073709551615 无符号*。在括号中规定最大位数。
FLOAT(size,d)	带有浮动小数点的小数字。在括号中规定最大位数。在 d 参数中规定小数点右侧的最大位数。
DOUBLE(size,d)	带有浮动小数点的大数字。在括号中规定最大位数。在 d 参数中规定小数点右侧的最大位数。

MySQL 数据类型

三．Date 类型：

数据类型	描述
DATE()	日期。格式：YYYY-MM-DD 注释：支持的范围是从 '1000-01-01' 到 '9999-12-31'
DATETIME()	*日期和时间的组合。格式：YYYY-MM-DD HH:MM:SS 注释：支持的范围是从 '1000-01-01 00:00:00' 到 '9999-12-31 23:59:59'
TIME()	时间。格式：HH:MM:SS 注释：支持的范围是从 '-838:59:59' 到 '838:59:59'
YEAR()	2 位或 4 位格式的年。 注释：4 位格式所允许的值：1901 到 2155。2 位格式所允许的值：70 到 69，表示从 1970 到 2069。
DATE()	日期。格式：YYYY-MM-DD 注释：支持的范围是从 '1000-01-01' 到 '9999-12-31'
DATETIME()	*日期和时间的组合。格式：YYYY-MM-DD HH:MM:SS 注释：支持的范围是从 '1000-01-01 00:00:00' 到 '9999-12-31 23:59:59'

第37章 数据库设计范式

华清远见-成都中心-H5教学部

数据库设计范式

目录

设计范式概述

数据库的设计范式是指数据库设计所需要满足的规范；只有理解数据库的设计范式，才能设计出高效、优雅、满足实际需求的数据库。

通俗理解数据库设计范式。

什么是通俗理解：三大范式是设计数据库的基本理念，可建立冗余较小、结构合理的数据库。如果有特殊情况，当然要特殊对待，数据库设计最重要的是看需求跟性能，需求>性能>表结构，不能一味的去追求范式建立数据库。

关系R：实质上是一张二维表，其中每一行是一个元素，每一列是一个属性。

第一范式

当关系模式R的属性不能再分解为更基本的数据单位时，保持了属性的原子性和唯一性，不产生冗余数据，则称R是满足第一范式，简称1NF；1NF是关系模式的最低要求。

第一范式:

① 每一列属性都是不可再分的属性值，确保每一列的原子性；

② 两列的属性相近或相似或一样，尽量合并属性一样的列，确保不产生冗余数据；

第二范式

若R满足1NF，且它的每一非主属性完全依赖于主键，从而保证一行数据只做一个事情，则称R满足第二范式，简称2NF。

第二范式:

在一个数据库表中，一个表中只能保存一种数据，一行数据只做一个事情。

不可以把多种数据保存在同一张数据库表中。

如果数据列中出现数据重复，我们需要考虑把表拆分开来。

第三范式

若R满足2NF，且每一非主属性不传递依赖于主键，也就是让非主属性和主键之间有直接关系，则称R满足第三范式，简称3NF。

第三范式:

数据不能存在传递关系，即每个属性都跟主键有直接关系而不是间接关系。。

海量视频 贴身学习

超多干货 实时更新

THANKS

谢谢